BÀI TẬP GIỮA KỲ

Môn: **XỬ LÝ ẢNH**

Giảng viên: TS. Nguyễn Thị Ngọc Diệp

Đề tài thực hiện: Finding Objects In Image

Mã sinh viên: 19020509

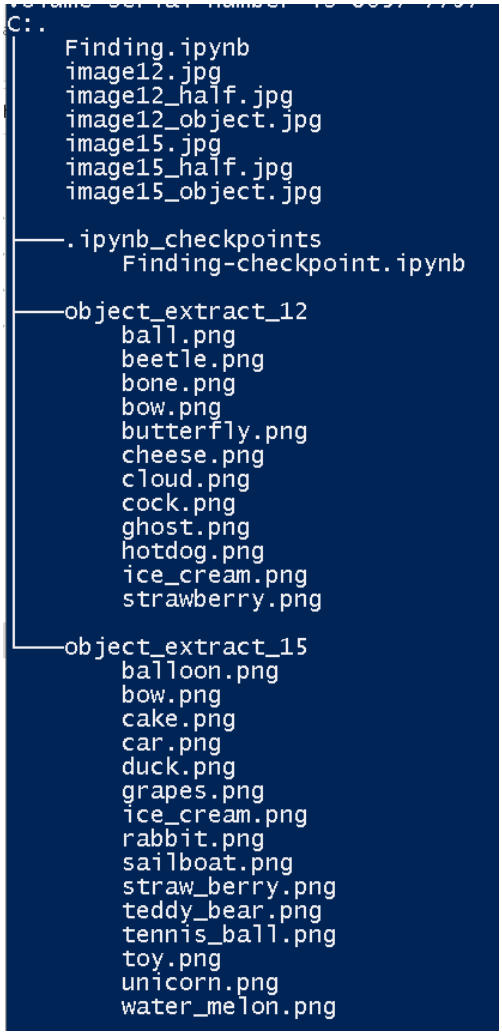
Họ và tên: Đỗ Nguyễn Cương

LMH: 2122I_INT3404_1

Mục lục:

I. SUBMISSION FOLDER STRUCTURE

Bài nộp dưới hình thức 01 folder. Bên trong bao gồm 01 file .jpynb của Jupyter Notebook chứa code thực thi chương trình. Các file hình ảnh *image12*, *image15* là những file ảnh tổng mà để đã cho. Các ảnh có thêm hậu tố *_half* là các ảnh đã cắt bỏ phần text cũnh như object mẫu. Các ảnh có hậu tố *_object* là ảnh cắt riêng phần object mẫu để phục vụ việc tách tự động. Ngoài ra, còn có 02 folder chứa các object đã được cắt ra từ *image12_object*, *image15_object* tương ứng với folder *object_extract_12* và *object_extract_15*.



II. BASIC PROGRAM

Tiếp cận với đề bài, đề yêu cầu tìm vật thể có trong ảnh dựa trên mẫu (template) có sẵn. Vì vậy, sau khi tìm hiểu, em quyết định sử dụng hàm **matchTemplate** là một hàm của thư viện OpenCV. Hàm **matchTemplate** yêu cầu các tham số là ảnh gốc chứa nhiều vật thể cần tìm và ảnh vật thể. Bước đầu em đã xử lý thủ công với các hình ảnh. Tách từ ảnh đề cho ra ảnh chỉ chứa các vật thể mẫu và ảnh chứa các vật thể đã xếp lộn xộn.

image15

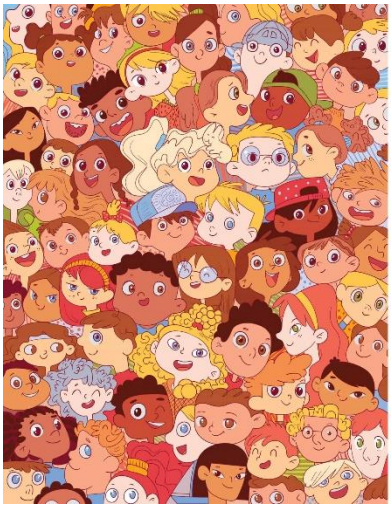


image15_half

image15_object

Tiếp theo, em đặt sẵn tên các object đượcs cắt ra thủ công theo tiếng anh để tiện cho các công việc sau này (Ví dụ: ice_cream, car, sailboat, balloon,…). Đến đây, công đoạn tiền xử lý coi như đã hoàn tất.

Bước vào phần code cho chương trình. Em **import các thư viện**:

- **numpy –** sử dụng các hàm tính toán
- **cv2 –** rất nhiều hàm sử dụng nhằm xử lý ảnh
- **matplotlib –** hiển thị hình ảnh (jupyter notebook không hỗ trợ hiển thị bằng cv2)
- **imultis –** chứa một số hàm thông dụng để xử lý hình ảnh cơ bản
- **time –** tính toán thời gian thực hiện chương trình (tuỳ chọn)

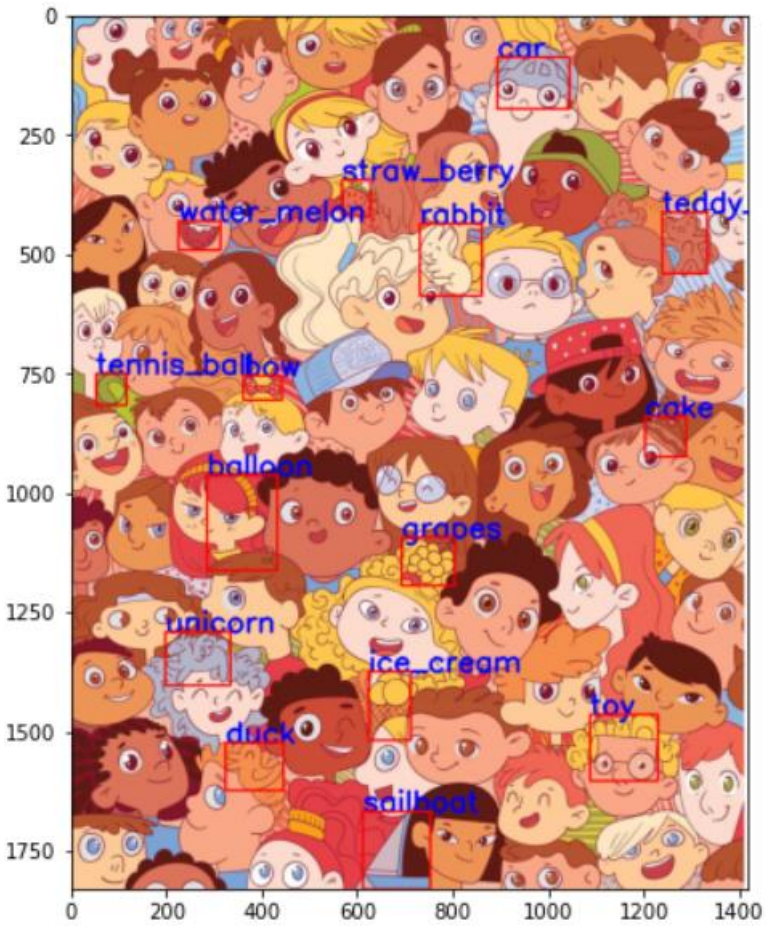
Tiếp theo là công đoạn **đọc hình ảnh**. Hình ảnh tổng để tìm vật thể là hình ảnh đã được cắt bỏ các phần không cần thiết, nhằm tránh gặp các vấn đề không cần thiết (*image15_half, image12_half*). Các vật thể được lưu trữ dưới dạng mảng trong Python. Sở dĩ em không lưu đầy đủ cả phần mở rộng của ảnh (*balloon* vs *balloon.png*,v.v..*)* là bởi phần danh sách nãy sẽ được tận dụng để hiển thị tên của các vật thể sau khi tìm được.

Vào phần **chương trình chính**. Em có sử dụng hàm **time** để tính thời gian thực thi chương trình, nên trước bắt đầu và kết thúc phần code **chương trình chính** sẽ có hai câu lệnh của hàm này. Cả phần chương trình này sẽ được bọc trong một vòng **for,** bởi vì với mỗi lần chạy, hàm **matchTemplate** chỉ có thể tìm được một vật thể, vậy để có thể tìm được tất cả các vật thể mà đề yêu cầu, ta cần sử dụng vòng for, số lượng lần lặp chính là số lượng object cần tìm.

Với mỗi vòng **for**, ta sẽ đọc từng object (template). Bởi vì hàm **matchTemplate** nguyên gốc có một đặc điểm, đó là nó sẽ chỉ cho ra kết quả tốt nhất khi mà template, có cùng kích cỡ, cùng hướng, cùng ánh sáng với vật thể đang nằm trong ảnh tổng. Chính bởi vì vậy, sau khi tìm hiểu, em đã cải tiến một chút, sử dụng kỹ thuật multi-scale, thật ra chỉ đơn giản là việc cho template scale tại lần lượt các mức khả thi, nhằm tìm ra mức scale đem lại kết quả tốt nhất. Hàm **resize** của thư viện **imutils** được e lựa chọn để resize ảnh, bởi hàm này có một điểm lợi đó là chỉ cần truyền vào kích thước một cạnh của hình ảnh cần resize mà vẫn giữ nguyên tỷ lệ ảnh. Và như đã nói, với mỗi mức scale, ta sẽ chạy **matchTemplate** một lần, phương thức tính được em lựa chọn sử dụng là TM_CCOFFF_NORMED. Với phương thức này, giá trị max của ma trận kết quả sẽ chính là nơi có độ matching lý tưởng nhất. Như vậy sau công đoạn này, ta đã có được giá trị lý tưởng nhất của mức scale này. Và như vậy, với mỗi lần scale, ta chỉ việc so sánh các giá trị lý tưởng này với nhau để tìm được giá trị cao nhất, cũng chính là lý tưởng nhất.

Sau khi đã tìm được điểm có giá trị có độ matching cao nhất, việc còn lại chỉ là vẽ bounded box xung quanh điểm đó. Em sử dụng hàm **rectangle** để vẽ bounded box, ngoài ra em còn kết hợp sử dụng hàm **putText** để hiển thị tên của các vật thể tìm được dựa vào data chính là mảng danh sách vật thể đã implement ở trên.

Sau khi hoàn tất các khâu xử lý, việc còn lại chỉ là show kết quả lên.

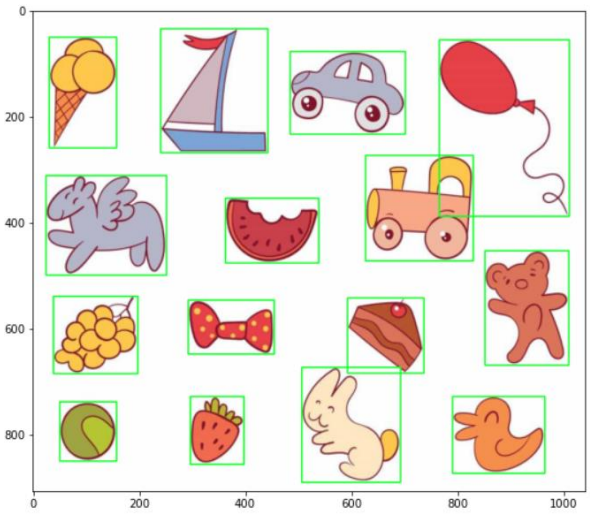


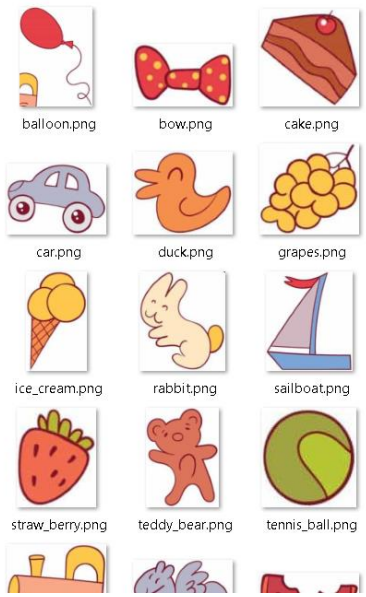
III. SOME IMPROVEMENTS

Chương trình trên cơ bản đã làm được nhiệm vụ, đó là tìm kiếm vật thể. Tuy nhiên, nó chỉ chạy hoàn hảo với bức ảnh có 12 object mà đề đã cho, vì bức ảnh này tương đối đơn giản. Tuy nhiên, khi áp dụng vào bức ảnh có 15 object, chương trình đã có những tính toán sai lệch. Chính bởi vì vậy, qua quá trình tìm hiểu thêm, em tiếp tục sử dụng kỹ thuật **Edge Detection** nhằm giúp hàm **matchTemplate** đem lại kết quả tối ưu hơn. Vì vậy em đã sử dụng hàm **Canny** để thực hiện kỹ thuật này. Hàm **Canny** được sử dụng cho cả hình ảnh

tổng và hình ảnh template. Và sau khi detect cạnh, chương trình đã chạy được cho cả 2 hình ảnh (image15, image12) và đem lại kết quả tốt.

Tiếp theo là một cải tiến khác, tại công đoạn **tiền xử lý**, nhận thấy việc cắt thủ công các vật thể mẫu ra làm template khá mất thời gian. Nên em đã tìm hiểu cách để có cắt tự động trích xuất các object này ra. Nhận thấy các object mẫu đều được đặt trên nền trắng (*imagexx_object*), qua tìm hiểu, em quyết định sử dụng kỹ thuật Contours Detection. Một số công đoạn tiền xử lý trước khi sử dụng hàm **findContours**, sử dụng Edge Detection kết hợp với blur ảnh bằng Gaussian Blur nhằm đem lại kết quả detect cạnh hoàn hảo nhất. Lúc này ảnh đã ở dạng binary. Tiếp tục sử dụng hàm **dilate** nhằm giãn, lấp đầy các phần nhỏ, khuyết trong ảnh, việc này sẽ giúp cho hàm **findContours** hoạt động hoàn hảo hơn. Sau khi có kết quả contours, việc còn này chỉ là vẽ bounded box và xuất hình ảnh đó ra. Hàm **boundingRect** sẽ cung cấp toạ độ điểm top-left và kích thước chiều dài, rộng dựa trên Contours. Khi đã có những thông tin này. Ta chỉ cần dùng hàm **rectangle** để vẽ bounded box, hoặc xuất trực tiếp hình ảnh (**imwrite)**.





Các object sau khi trích xuất ra sẽ theo một định dạng tên nhất định. Em đổi tên thủ công.

Bài tập còn rất nhiều điểm cần cải thiện, em sẽ tiếp tục tìm hiểu để chương trình hoạt động tối ưu hơn. Em xin cảm ơn cô vì đã đưa ra những đề tài thật sự rất thú vị, cũng như tạo điều kiện về thời gian cho chúng em. Kính chúc cô có nhiều sức khoẻ.